Example of Tuning Cube II

Date: Dicember 2016

Author: Ramón Portolés, Alberto Linkedin a.ramonportoles@gmail.com

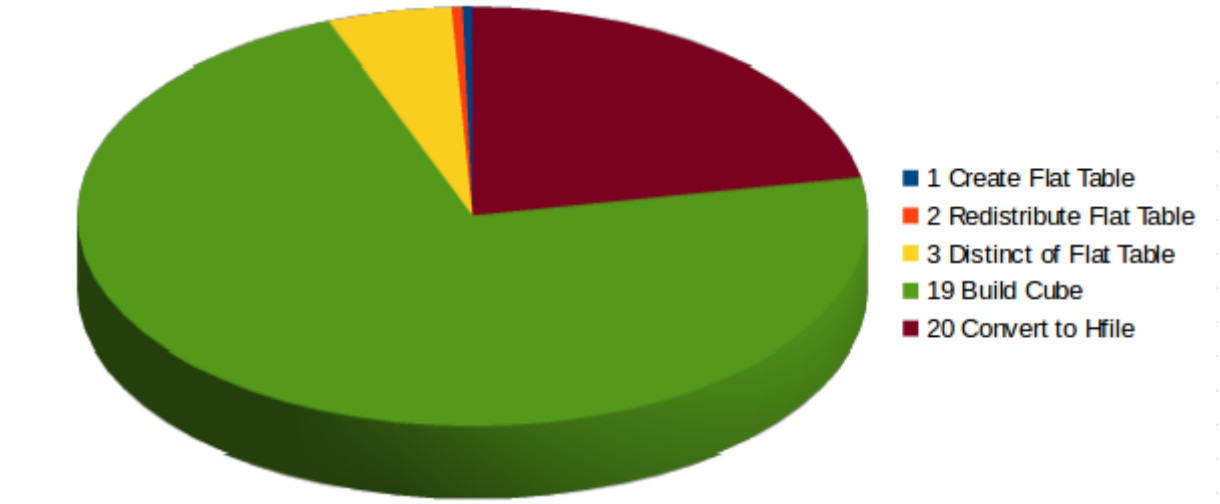
The tunning process has been

- Hive Input tables compressed
- HBase Output compressed
- Apply techniques of reduction of cardinality (Joint, Derived, Hierarchy and Mandatory)
- Personalize Dim encoder for each Dim and choose the best order of Dim in Row Key

Now, there are three types of cubes:

- Cubes with low cardinality in their dimensions (Like cube 5, most of time is usend in flat table steps)
- Cubes with high cardinality in their dimensions (Like cube 7,most of time is usend on Build cube, the flat table steps are lower than 10%)
- The third type, ultra high cardinality (UHC) which is outside the scope of this article

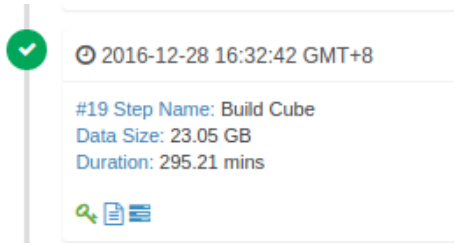
Cubes 7: Cube with high cardinality Dimensions



In this case **72%** of the time is used to build Cube

This step is a MapReduce task, you can see the Sql Hive of these steps on Monitor > 

How can the performance of Map – Reduce be improved? The easy way is to increase the numbers of Mappers and Reduces (= Increase parallelism).



NOTE: YARN / MapReduce have a lot parameters to configure and adapt to the system. The focus here is only on small parts.

(In my system I can assign 12 – 14 GB and 8 cores to YARN Resources):

yarn.nodemanager.resource.memory-mb = 15 GB

- yarn.scheduler.maximum-allocation-mb = 8 GB
- yarn.nodemanager.resource.cpu-vcores = 8 cores

With this config our max theoretical grade of parallelism is 8. However, this has a problem: “Timed out after 3600 secs”

Attempt	State	Elapsed Time	Note
attempt_1482868137604_0009_m_000000_0	FAILED	1hrs, 57sec	AttemptID:attempt_1482868137604_0009_m_000000_0 Timed out after 3600 secs
attempt_1482868137604_0009_m_000000_1	FAILED	1hrs, 56sec	AttemptID:attempt_1482868137604_0009_m_000000_1 Timed out after 3600 secs

The parameter *mapreduce.task.timeout* (1 hour by default) define max time that Application Master (AM) can happen with out ACK of Yarn Container. Once this time passes, AM kill the container and retry the same 4 times (with the same result)

Where is the problem? The problem is that 4 mappers started, but each mapper needed more than 4 GB to finish

- The solution 1: add more RAM to YARN
- The solution 2: increase vCores number used in Mapper step to reduce the RAM used
- The solution 3: You can play with max RAM to YARN by node (*yarn.nodemanager.resource.memory-mb*) and experiment with min RAM per container (*yarn.scheduler.minimum-allocation-mb*). If you increase minimum RAM per container, YARN will reduce the numbers of Mappers

Show 20 ▾ entries

Task	Progress	
task_1482877261039_0005_m_000000		hdfs://amb0.m 72d53d0ee2d3 > map
task_1482877261039_0005_m_000001		hdfs://amb0.m 72d53d0ee2d3 > map
task_1482877261039_0005_m_000002		hdfs://amb0.m 72d53d0ee2d3 > map
task_1482877261039_0005_m_000003		NEW

In the last two cases the result is the same: reduce the level of parallelism ==>

- Now 3 mappers start at the same time, the fourth must wait for a free slot
- The three first mappers distribute the ram among themselves, and as a result they will have enough ram to finish the task

During a normal “Build Cube” step you will see similar messages on YARN log:

```
[CuboidTask-2] org.apache.kylin.cube.inmemcubing.InMemCubeBuilder: Cuboid 53059 has 948266 rows, build takes 1219ms
[CuboidTask-2] org.apache.kylin.cube.inmemcubing.InMemCubeBuilder: collecting CuboidResult cuboid id:53059
[CuboidTask-2] org.apache.kylin.cube.inmemcubing.InMemCubeBuilder: Calculating cuboid 52995
[CuboidTask-2] org.apache.kylin.gridtable.GTScanRequest: pre aggregating results before returning
[CuboidTask-3] org.apache.kylin.cube.inmemcubing.InMemCubeBuilder: Cuboid 118531 has 955928 rows, build takes 12667ms
[CuboidTask-3] org.apache.kylin.cube.inmemcubing.InMemCubeBuilder: collecting CuboidResult cuboid id:118531
[CuboidTask-3] org.apache.kylin.cube.inmemcubing.InMemCubeBuilder: Calculating cuboid 53057
[CuboidTask-3] org.apache.kylin.gridtable.GTScanRequest: pre aggregating results before returning
[CuboidTask-0] org.apache.kylin.cube.inmemcubing.InMemCubeBuilder: Cuboid 118593 has 938836 rows, build takes 11976ms
[CuboidTask-0] org.apache.kylin.cube.inmemcubing.InMemCubeBuilder: collecting CuboidResult cuboid id:118593
[CuboidTask-0] org.apache.kylin.cube.inmemcubing.InMemCubeBuilder: Calculating cuboid 53058
[CuboidTask-0] org.apache.kylin.gridtable.GTScanRequest: pre aggregating results before returning
```

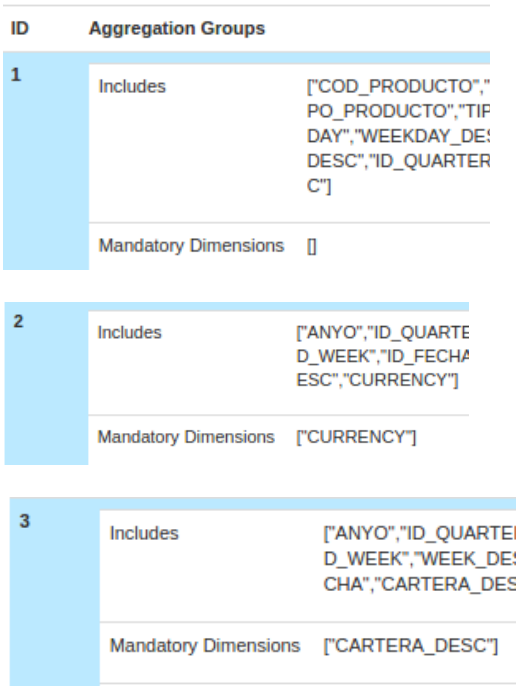
If you don’t see this periodically, perhaps you have a bottleneck in the memory

Cube 7: Improve cube response time

We can try to use different aggregation groups to improve the query performance of some very important Dim or a Dim with high cardinality.

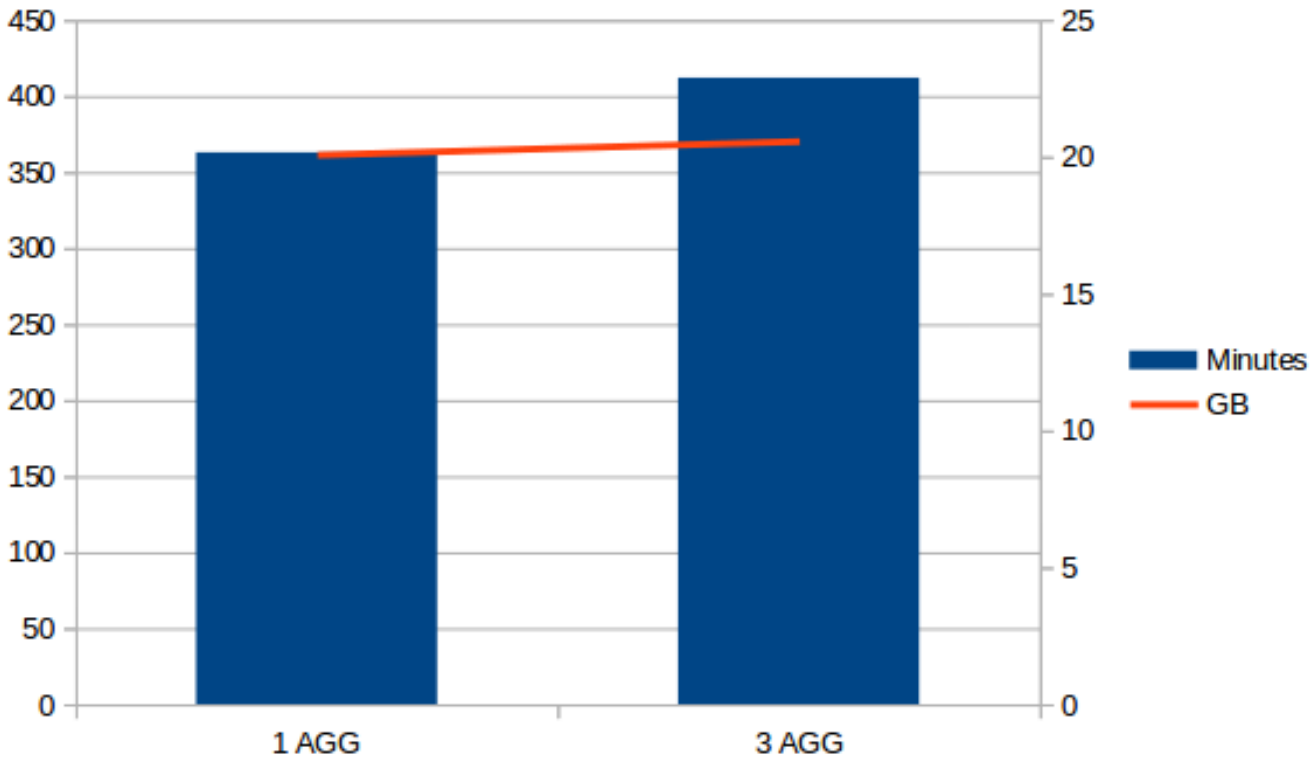
In our case we define 3 Aggregation Groups:

1. “Normal cube”
2. Cube with Date Dim and Currency (as mandatory)
3. Cube with Date Dim and Carteras_Desc (as mandatory)



ID	Aggregation Groups	
1	Includes	["COD_PRODUCTO","PO_PRODUCTO","TIP DAY","WEEKDAY_DES DESC","ID_QUARTER C"]
	Mandatory Dimensions	[]
2	Includes	["ANYO","ID_QUARTE D_WEEK","ID_FECHA ESC","CURRENCY"]
	Mandatory Dimensions	["CURRENCY"]
3	Includes	["ANYO","ID_QUARTE D_WEEK","WEEK_DES CHA","CARTERA_DES
	Mandatory Dimensions	["CARTERA_DESC"]

Compare without / with AGGs:



Now it uses 3% more time to build the cube and 0.6% of space, but queries by currency or Carteras_Desc will be much faster.